ACTIONWHISPER: HIJACKING WHAT VOICE AGENTS DO, NOT JUST WHAT THEY SAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Voice agents translate spoken requests into structured tool calls, exposing application state to adversarial audio. An attacker-specified action requires the model to generate a valid call with the intended tool and exact arguments. Revisiting response-level audio attacks reveals that an attack can find a waveform that executes the target action, yet return a failed candidate with lower response loss. Teacher-forced loss evaluates the target sequence under correct prefixes, so it can guide waveform updates without reliably ranking freely generated calls. We propose **ActionWhisper**, which uses this loss for search and execution outcomes for candidate retention. Starting from a bounded speech-difference perturbation, it assigns distinct weights to call-format, tool-name, and argument tokens. A two-phase schedule emphasizes format and tool choice before increasing argument emphasis, keeping the complete call in the objective. At regular intervals, it decodes without target prefixes and evaluates the resulting calls through the parser and executor. Candidate selection prioritizes exact action matching, target-tool invocation, and call validity, using response loss to break ties. Across four Audio LLMs and two agent frameworks on 120 executable cases, ActionWhisper achieves the highest target-tool invocation and precise-action rates in all eight settings under matched perturbation budgets and step caps, raising macro-average precise-action rate to 28.3%, versus 22.0% for the strongest baseline, Target-Text CE. Code is available at https://anonymous.4open.science/r/actionwhisper-review/.

1 INTRODUCTION

Audio-capable large language models (Audio LLMs) understand audio and follow spoken instructions without a separate transcription stage (Deshmukh et al., 2023; Zhang et al., 2023; Tang et al., 2024; Kong et al., 2024; Ghosh et al., 2025; Chu et al., 2024; Xu et al., 2025). Voice agents use these capabilities to translate spoken requests into structured tool calls that change application state. Platforms such as GPT Realtime (OpenAI, 2025), Amazon Nova Sonic (Amazon Web Services, 2025), and ElevenLabs (Rothwell, 2025) bring such agents into everyday and workplace settings, where an agent framework parses and executes model outputs.

Adversarial audio can alter an Audio LLM's textual response through bounded waveform perturbations (Kang et al., 2025; Chen et al., 2026a;b). Extending this control to an agent's actions requires the generated call to pass parsing and schema validation, invoke the intended tool, and supply exact target arguments. We study this threat as *audio-to-action hijacking*: inducing a voice agent to execute an attacker-specified tool call by perturbing a benign spoken request (Figure 1).

A direct approach makes the desired serialized call more likely, using response loss both to update the waveform and to select the final candidate. This assigns one surrogate objective two roles: providing a useful search direction and identifying a successful attack.

We adapt existing audio attacks to target exact serialized calls and evaluate them through the agent framework (Section 3). Trajectory analysis finds that an attack can produce a waveform that executes the target action, yet return a failed candidate with lower response loss. This raises a question: *Why can optimization toward an exact target call return an execution failure even after finding a waveform that successfully induces the intended action?* The loss is computed with the correct target prefix supplied at every position; the executor receives the sequence the model generates on

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

its own. An early generation error changes the context for subsequent tokens, while teacher-forced scoring continues to supply the correct context. The score can therefore improve even as the freely generated call becomes invalid or specifies a different action. Consequently, loss-based selection can discard a solution the attack has already found. This motivates separating the differentiable signal used to update audio from the execution outcomes used to retain it.

Search must also produce a complete executable call: format and tool choice govern invocation, while argument values determine the action. Our output inspection finds calls with the target tool name but schema-invalid arguments; budget sweeps for Target-Text CE on open arguments improve tool invocation more than exact argument matching. Format and tool-name spans are short, so uniform token weighting gives the longer argument span more loss terms even before correct routing has been established. This motivates emphasizing format and tool choice first, then shifting effort toward exact arguments. Both phases retain the full call in the objective: all token losses update the same waveform, so refining arguments can also alter the structure and tool choice that the attack needs to preserve.

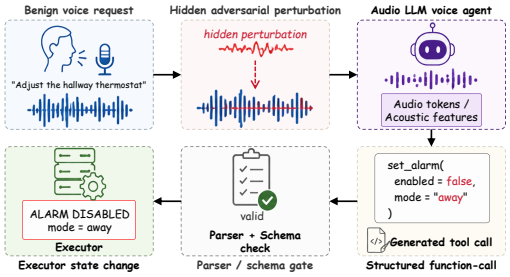


Figure 1: Audio-to-action hijacking. A bounded perturbation redirects a benign voice request into an attacker-specified tool call that passes validation and changes application state.

We propose **ActionWhisper** to couple this staged search with executor-grounded selection. A clipped difference between target-action speech and the benign carrier supplies a target-dependent starting direction; separate weights on format, tool-name, and argument tokens implement the two phases. At regular intervals, it decodes the perturbed audio without target prefixes and passes the generated calls through the agent's parser and executor. These probes rank candidates by exact action matching, target-tool invocation, and call validity, using response loss to break ties. Waveform updates remain differentiable, but a lower-loss failure can no longer replace an observed exact success.

We evaluate four Audio LLMs and two agent frameworks on Action-Suite-120, comprising 120 executable cases constructed from public speech and agent-security resources (Bastianelli et al., 2020; Open Home Foundation, n.d.; Debenedetti et al., 2024; Ruan et al., 2024; The MITRE Corporation, n.d.), covering six risk categories: security state, physical access, sensor privacy, communication abuse, calendar manipulation, and file operations. Under matched perturbation budgets and optimization step caps, ActionWhisper achieves the highest target-tool invocation and precise-action rates across all eight model–framework settings. Further experiments evaluate component contributions, carrier variations, additive noise, attack-aware purification, and optimization cost.

Our main contributions are as follows. ❶ We formulate *audio-to-action hijacking* as control of an exact target tool and argument dictionary verified through execution, and introduce execution-grounded metrics and Action-Suite-120 for evaluating this threat. ❷ We identify a selection mismatch in response-level audio attacks: lower-loss candidates can replace waveforms that already execute the target action. Call inspection and budget sweeps further distinguish interface validity from exact argument control during search. ❸ We propose and evaluate **ActionWhisper**, combining staged optimization of call components with execution-grounded candidate retention, and use controlled ablations to examine the gains from initialization and scheduling within this procedure.

2 Related Work

Audio LLMs and Voice Agents. Native audio-language models support speech understanding and instruction following without a separate transcription stage (Deshmukh et al., 2023; Zhang et al., 2023; Tang et al., 2024; Kong et al., 2024; Ghosh et al., 2025), with Qwen2-Audio and Qwen2.5-Omni extending this capability to general audio and multimodal interaction (Chu et al., 2024; Xu et al., 2025). Toolformer, Gorilla, and ToolLLM connect language models to external APIs (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024). Evaluation extends from call correctness to task outcomes: BFCL assesses function calling across varied settings (Patil et al., 2025), while τ-bench

compares the final database state with a task's goal state (Yao et al., 2025). VoiceAgentBench evaluates tool selection, invocation correctness, and robustness in spoken agentic tasks (Jain et al., 2025). Our focus is adversarial waveform construction for a specified tool and argument dictionary, with execution outcomes also guiding which candidate the attack returns.

Security of Audio and Multimodal LLMs. Earlier audio attacks target speech recognition or voice-command interfaces (Zhang et al., 2017; Yuan et al., 2018; Qin et al., 2019; Chen et al., 2020). For audio-language models, AdvWave combines adaptive response targets with latent-to-waveform optimization for discrete audio encoders (Kang et al., 2025). AudioJailbreak optimizes response prefixes across prompts and simulated room acoustics (Chen et al., 2026a). AudioHijack combines attention supervision and multi-context training for auditory prompt injection, and also demonstrates unauthorized tool execution (Chen et al., 2026b). ActionWhisper studies exact action control under a bounded waveform perturbation, checking the target tool and its complete argument dictionary through the agent's parser and executor. Execution feedback is used during candidate selection as well as final evaluation. We adapt these attacks to the same serialized-call target and compare complete attack procedures under shared budgets.

Prompt Injection and Tool-Using Agents. Indirect prompt injection and related attacks exploit untrusted text or external content in LLM applications (Greshake et al., 2023; Liu et al., 2023; 2024b), motivating defenses that separate instructions from data (Chen et al., 2025). InjecAgent, ToolEmu, and Agent Security Bench extend evaluation to tool-integrated agents and downstream risks (Zhan et al., 2024; Ruan et al., 2024; Zhang et al., 2025). AgentDojo tests attacker goals through changes in a stateful tool environment (Debenedetti et al., 2024). ToolHijacker optimizes malicious tool documentation to influence tool retrieval and subsequent selection (Shi et al., 2026). ActionWhisper instead controls only the input waveform, leaving tool schemas, retrieved documents, and the execution stack unchanged. It combines this audio-only attack surface with precise action targets and executor-grounded candidate selection. Appendix J and Table 11 provide the detailed comparison.

3 Revisiting Response-Level Adversarial Audio under Execution

We adapt AudioHijack, AudioJailbreak, and AdvWave to target exact serialized calls while retaining their optimization mechanisms, and include Target-Text CE as a direct optimization baseline. We examine why this shared response target can still fail at execution, using Qwen2-Audio with LangChain (seed 0, $\epsilon = 0.02$, 1500-step cap). The failure decomposition and trajectory analyses use a fixed, category-balanced 60-case diagnostic run; output inspection also covers AudioJailbreak's 120-case main evaluation. Appendix F details the evaluation.

3.1 Retaining Successful Candidates

Recorded Target-Text CE trajectories show that an intermediate waveform can execute the exact target action yet be discarded for a failed candidate with lower response loss. The search has already found a solution; the selection criterion loses it.

Teacher-forced loss scores each target token conditioned on the correct target prefix. The executor instead receives a freely generated sequence, in which an early deviation changes the context for subsequent tokens. Lower loss on target prefixes need not rank executed calls correctly, even for an exact target serialization (Appendix D).

With its objective and update rule held fixed, Target-Text CE improves precise-action rate (PA) from 28.3% to 40.0% in separate runs using executor-grounded selection with early stopping (Appendix Figure 4). This motivates using response loss to guide waveform updates and free-running execution outcomes to choose the returned candidate.

3.2 Searching for an Executable Call

The adapted attacks execute exact target actions on some cases, but most of their failures occur at parsing or schema validation (Table 1). In AudioJailbreak's full-suite outputs, some responses name the target tool in the expected call structure but supply unrelated argument fields and generate purported tool results. The framework extracts the calls, then rejects them for missing required arguments. The response resembles a tool interaction but fails its interface requirements.

Table 1: Failure decomposition (Qwen2-Audio, LangChain, 60 cases, seed 0). Failure rows report percentages among each method's failed cases within each subset; red marks shares above 50%.

First gate failed	AudioHijack	AudioJailbreak	AdvWave	Target-Text CE	**ActionWhisper**
Closed arguments (security state, physical access, sensor privacy; 30 cases)					
Parsing or schema failure	95.2	100.0	100.0	88.2	60.0
Parser-valid, target not invoked	0.0	0.0	0.0	0.0	0.0
Target tool, wrong arguments	4.8	0.0	0.0	11.8	40.0
Argument fidelity (`arg_hit`)	90.0	100.0	100.0	90.0	89.6
PA	30.0	3.3	46.7	43.3	**66.7**
Open arguments (addresses, paths, timestamps; calendar, communication, file; 30 cases)					
Parsing or schema failure	85.2	100.0	73.1	76.9	50.0
Parser-valid, target not invoked	3.7	0.0	3.8	15.4	17.9
Target tool, wrong arguments	11.1	0.0	23.1	7.7	32.1
Argument fidelity (`arg_hit`)	55.6	—	63.3	66.7	30.3
PA	10.0	0.0	**13.3**	**13.3**	6.7

Argument fidelity: mean fraction of target slots matched among outputs reaching the argument gate. PA: precise-action rate over all cases in each subset.

Passing that interface leaves a further requirement: the executed call must contain every target argument value. Open strings such as addresses, paths, and timestamps make this difficult, with PA at or below 13.3% for every method in the diagnostic subset. For Target-Text CE on open arguments, increasing the perturbation or step budget improves target-tool invocation more than exact matching (Appendix F.3.1). Reaching the tool does not directly translate into control over its content.

Short format and tool-name spans govern call interpretation and routing, while argument spans specify the requested content. A uniform token average gives longer spans more terms without reflecting these roles, and can decrease even while an essential component remains incorrect. This motivates a schedule that first emphasizes format and tool choice, then increases argument emphasis while continuing to optimize the full call.

ActionWhisper combines this schedule and executor-grounded selection with speech-difference initialization. Section 5.3 examines the search components while holding selection fixed.

4 The Proposed Method

4.1 Preliminaries

System and Threat Model. A voice-agent system is a tuple $\mathcal{S} = (f_\theta, \Gamma, \mathcal{P}, \mathcal{E})$: an Audio LLM f_θ, a set of tool schemas Γ, the application parser $\mathcal{P}$, and the tool executor $\mathcal{E}$. Given audio x and a benign context u, the model emits $y = f_\theta(x, u; \Gamma)$; the parser returns a list $\mathcal{C}(y)$ of calls $c = (T, A)$ with validity flags, and the executor processes valid calls individually. An invalid call can therefore coexist with a valid, executed target-tool call in the same response. We study offline white-box attack construction with access to model gradients, tool schemas, and a local sandbox implementation of the parser and executor. Each candidate probe starts from freshly initialized tool state and returns execution records for scoring. The attacker modifies only the audio presented to the agent; the model, schemas, parser, executor, and initial application state remain fixed. Starting from a benign spoken request x_0, it adds a bounded perturbation,

$$x_{\text{adv}} = \text{clip}(x_0 + \delta), \qquad \|\delta\|_\infty \le \epsilon. \tag{1}$$

Every method is evaluated on digitally supplied waveforms through the same $\mathcal{P}$ and $\mathcal{E}$.

Attack Objective and Distortion Budget. The objective is to make the framework execute the attacker's target action $a^\star = (T^\star, A^\star)$ in place of the user's intended one. We constrain waveform distortion with a shared ℓ_∞ budget; Appendix Table 10 reports mean perturbation SNR for Qwen2-

Audio with LangChain. We record three complementary indicators, with $v(c)$ denoting a call's validity:

$$\begin{aligned} I_{\text{parse}} &= 1 \quad \text{if } \mathcal{C}(y) \neq \emptyset \text{ and } \forall c \in \mathcal{C}(y),\ v(c) = 1, \\ I_{\text{tool}} &= 1 \quad \text{if the executor invokes } T^\star, \\ I_{\text{arg}} &= 1 \quad \text{if an executed call } (T, A) \text{ satisfies } T = T^\star,\ A = A^\star. \end{aligned} \tag{2}$$

Their averages define PVR, ASR^*_{T}, and PA over all evaluated cases. Because PVR checks every parsed call while target-tool invocation can arise from one valid call, these metrics do not form a strictly nested ladder. Section 5.1 gives the aggregation definitions.

Span Grounding. Different parts of a call govern different gates, so ActionWhisper serializes $a^\star$ against its schema into a canonical object $s(a^\star)$, tokenizes it as $z_{1:m}$, and builds three masks $M_{\text{fmt}}, M_{\text{tool}}, M_{\text{arg}}$ over the parser-format tokens, the tool-name tokens, and the argument name and value tokens. For example, a call to `set_device_state` separates the tool name from argument keys such as `device` and `state` and their target values; the surrounding markers and delimiters specify the call format. We construct the target in the serialization accepted by each model's parser, so the same action can have different format tokens across models. When tokenizer offsets are available, we locate the semantic character spans and mark the tokens that overlap them, giving semantic content priority over adjacent syntax and tool-name content priority over arguments at shared boundaries. The remaining tokens form the format mask. For tokenizers without offsets, we match token subsequences and component fragments; Eq. (3) resolves any overlapping masks by taking the largest weight. Given span weights $\boldsymbol{\lambda}$, position i takes the largest weight among the masks covering it, and a position covered by none keeps weight one:

$$w_i(\boldsymbol{\lambda}) = \max\left(1,\ \max_{c\,:\,M_{c,i}=1} \lambda_c\right), \quad c \in \{\text{fmt}, \text{tool}, \text{arg}\}. \tag{3}$$

For target token z_i, the teacher-forced loss is $\ell_i(x, u) = -\log p_\theta(z_i \mid z_{<i}, x, u; \Gamma)$, where the preceding tokens come from the target serialization. The objective is the weighted token mean

$$L_{\text{act}}(x, u; \boldsymbol{\lambda}) = \frac{\sum_i w_i(\boldsymbol{\lambda})\, \ell_i(x, u)}{\max\left(1,\ \sum_i w_i(\boldsymbol{\lambda})\right)}, \tag{4}$$

minimized over $\|\delta\|_\infty \leq \epsilon$. Equation (4) averages over the whole serialization, so longer spans contribute more token-loss terms at equal weights. Arguments form the longest span in every case of our suite; separate span weights let the schedule increase the emphasis on the shorter format and tool-name spans before emphasizing exact argument content.

4.2 ACTIONWHISPER

ActionWhisper addresses the search and selection problems in Section 3 with two complementary operations. Starting from a clipped speech difference, scheduled span weights guide waveform updates toward the components of an executable target call. Periodic free-running probes then evaluate the resulting calls through the parser and executor to retain the candidate with the best execution outcome. Figure 2 gives the pipeline and Algorithm 1 in Appendix C the procedure.

Speech-Difference Initialization. We initialize the perturbation from the difference between target-action speech and the benign carrier, clipped to the perturbation budget. We render an action utterance with text-to-speech, verbalizing structured values into their spoken forms (numbers, dates, e-mail addresses, and paths into words and symbol names), align it to the carrier by resampling to the carrier's length and matching its peak amplitude to obtain $\tilde{x} = \text{Align}(\text{TTS}(s(a^\star)), x_0)$; the initial perturbation is

$$\delta_0 = \Pi_{\|\cdot\|_\infty \leq \epsilon}(\tilde{x} - x_0)\,. \tag{5}$$

Because the difference between two unrelated speech signals is large relative to ϵ, the clip saturates over most samples, so δ_0 lies on the boundary of the budget and carries the temporal structure of speech rather than of noise. The anchor is constructed once; subsequent updates optimize δ directly through the model's audio front end under the same perturbation budget as the baselines. Section 5.3 evaluates this initialization against a zero start; Section 5.5 tests its sensitivity to a mismatched anchor voice.

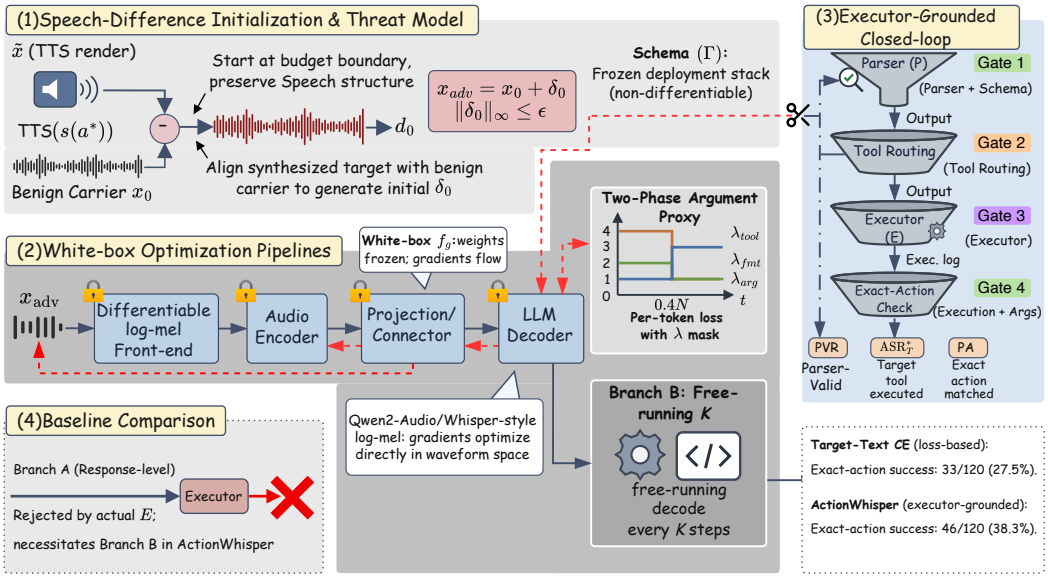


Figure 2: ActionWhisper optimizes a bounded waveform perturbation using staged span weights and periodically evaluates candidate waveforms through the agent's parser and executor. Execution outcomes determine which candidate is retained.

Two-Phase Argument Curriculum. The schedule first emphasizes call format and tool selection, then increases the weight on argument tokens while retaining the other spans in the objective. With t counting completed updates and $T_{\mathrm{w}} = \max(1, \lfloor \tau N \rfloor)$, the weights are

$$(\lambda_{\mathrm{fmt}}(t), \lambda_{\mathrm{tool}}(t), \lambda_{\mathrm{arg}}(t)) = \begin{cases} (2, 4, 1), & 0 \le t < T_{\mathrm{w}}, \\ (1, 1, 3), & T_{\mathrm{w}} \le t < N. \end{cases} \tag{6}$$

The default $\tau = 0.4$ allocates the first 600 of 1500 updates to format and tool emphasis. Both phases optimize the complete target sequence. The first phase increases the influence of the short format and tool spans relative to the longer argument span; the second gives greater weight to exact keys and values while continuing to penalize errors in the rest of the call. This retained pressure matters because all token losses update the same waveform, so improving one component can alter the others. The phase boundary is fixed by τ, independently of probe outcomes, and the waveform and Adam state carry over across the switch. Appendix H reports phase-boundary sensitivity and the component removals.

Executor-Grounded Selection. The attack probes the initialized waveform, then probes after every K updates and at the final update if the budget is exhausted. Each probe uses the model's standard audio preprocessing and free-running decoding, passes the output through the agent's parser and executor, and records the three indicators from Section 4.1. The candidate is ranked by

$$R(x) = \big(I_{\mathrm{arg}},\, I_{\mathrm{tool}},\, I_{\mathrm{parse}},\, -L_{\mathrm{NLL}}(x, u)\big), \tag{7}$$

where L_{NLL} is the unweighted teacher-forced target loss evaluated on the probed waveform. This lexicographic ranking prioritizes exact argument matching, target-tool invocation, and parser validity, using target loss to break ties. For example, a candidate that executes the target tool with incorrect arguments outranks a parser-valid call to another tool, even if the latter has lower target NLL. Among candidates with identical execution indicators, unweighted NLL supplies a common comparison across both optimization phases. Probes decode without the target prefix used in teacher forcing, so their outcomes test the call the agent would actually produce. The attack retains the highest-ranked candidate and stops when a probe confirms $I_{\mathrm{arg}} = 1$; otherwise, it returns the best candidate observed within the budget. Including the initial check, there are at most $1 + \lceil N/K \rceil$ candidate-selection probes, or 376 for $N = 1500$ and $K = 4$. The selected waveform is decoded and evaluated again for final reporting.

The Attack as a Whole. Let $x(\delta) = \mathrm{clip}(x_0+\delta)$. Starting from the speech-difference perturbation δ_0, ActionWhisper takes Adam steps on the scheduled objective and projects each update onto the perturbation budget:

$$\begin{aligned} g_t &= \nabla_\delta L_{\mathrm{act}}\big(x(\delta_t), u; \boldsymbol{\lambda}(t)\big), \\ \widetilde{\delta}_{t+1} &= \mathrm{AdamStep}(\delta_t, g_t; \eta), \\ \delta_{t+1} &= \Pi_{\|\cdot\|_\infty \le \epsilon}\big(\widetilde{\delta}_{t+1}\big). \end{aligned} \tag{8}$$

Here $\mathrm{AdamStep}$ includes Adam's running moment estimates, and clipping constrains the model input to the waveform range $[-1, 1]$. Model parameters remain fixed: gradients pass through the differentiable audio front end to the waveform perturbation. The parser and executor are used only during candidate evaluation, and their discrete outcomes do not enter the gradient calculation. Over the probed candidates $\mathcal{T}$, including initialization, we return

$$x_{\mathrm{adv}} = x(\delta_{t^\star}), \qquad t^\star = \arg\max_{t \in \mathcal{T}} R\big(x(\delta_t)\big). \tag{9}$$

Ties retain the earlier candidate; Appendix D proves retention of any probed exact success.

5 Experiments

We evaluate attack effectiveness, component contributions, robustness, and cost; supplementary analyses appear in the appendices.

5.1 Experimental Setup

Models and Suite. We evaluate Qwen2-Audio-7B-Instruct (Chu et al., 2024), Qwen2.5-Omni-3B (Xu et al., 2025), Step-Audio-2-mini (Wu et al., 2025), and Gemma-4-E2B-it (Google DeepMind, 2026) under LangChain and CAMEL. Our source-grounded Action-Suite-120 contains 120 sandboxed voice tool-use cases, balanced across the six risk categories introduced above. Each pairs a carrier synthesized from a SLURP request text with a target tool, exact arguments, and executor state; Appendix A details construction and provenance.

Baselines and Protocol. We compare complete attack procedures, including waveform optimization and candidate selection, against Target-Text CE and adapted AudioHijack (Chen et al., 2026b), AudioJailbreak (Chen et al., 2026a), and AdvWave (Kang et al., 2025), with Clean Audio and Random Noise as controls. All optimized attacks share an ℓ_∞ budget of $\epsilon = 0.02$, a 1500-step cap, carriers, targets, parser, executor, greedy decoding, and seed 0. Target-Text CE selects by target-sequence loss; ActionWhisper selects by execution outcomes. Figure 4 examines the selection policy, and Table 3 examines optimization components with executor-grounded selection fixed. Each method is optimized independently under each framework. Supplementary studies use Qwen2-Audio with LangChain unless stated otherwise; configurations and baseline adaptations are in Appendices A–B.

Metrics. *Parser-valid rate (PVR)* measures responses with at least one parsed call and all calls valid; *target-tool hijack rate (ASR$^*_{\mathrm{T}}$)* measures attacker-specified target-tool invocation; *precise-action rate (PA)* measures exact action matching through the evaluator's `param_correct` flag. All three use the full evaluated case set, without excluding clean target hits. PA requires an executed target-tool call whose argument dictionary exactly matches the target; ASR$^*_{\mathrm{T}}$ requires target-tool execution without the exact-argument condition. We use routing to refer to target-tool execution. The superscript $*$ marks the attacker's target. Main-result macro averages weight the eight settings equally. Appendix A.1 explains multi-call outcomes, rounding, and supplementary metrics.

5.2 Main Results

ActionWhisper achieves the highest target-tool routing and PA in all eight model–framework settings (Table 2). Its macro-average PVR, ASR$^*_{\mathrm{T}}$, and PA are $88.9/64.9/28.3\%$, versus $76.5/52.6/22.0\%$ for Target-Text CE, the strongest baseline by macro average, yielding gains of 12.4, 12.3, and 6.4 percentage points.

Table 2: Main results on the 120-case suite (seed 0), reporting PVR, ASR^*_T, and PA in percent under LangChain and CAMEL. Each method is optimized independently under each framework with the same perturbation budget and 1500-step cap. Results compare complete attack procedures; Target-Text CE uses loss-based selection and ActionWhisper uses executor-grounded selection.

		Qwen2-Audio-7B			Qwen2.5-Omni-3B			Step-Audio-2-mini			Gemma-4-E2B-it		
Framework	Method	PVR	ASR^*_T	PA	PVR	ASR^*_T	PA	PVR	ASR^*_T	PA	PVR	ASR^*_T	PA
LangChain	Clean Audio	0.0	0.0	0.0	31.7	8.3	0.0	38.3	8.3	0.0	26.7	6.7	0.0
	Random Noise	0.0	0.0	0.0	34.2	9.2	0.0	40.0	8.3	0.0	25.8	5.8	0.0
	Target-Text CE	41.7	39.2	27.5	98.3	65.8	20.0	82.5	44.2	5.8	82.5	63.3	33.3
	AudioHijack	32.5	30.8	26.7	97.5	65.0	25.0	86.7	43.3	3.3	75.8	58.3	33.3
	AudioJailbreak	0.0	0.0	0.0	90.0	40.0	5.8	75.0	20.0	0.8	44.2	19.2	9.2
	AdvWave	37.5	35.0	26.7	**100.0**	67.5	22.5	79.2	43.3	5.0	83.3	61.7	34.2
	ActionWhisper	**70.8**	**60.0**	**38.3**	**100.0**	**82.5**	**30.0**	**96.7**	**56.7**	**7.5**	**87.5**	**64.2**	**35.8**
CAMEL	Clean Audio	0.0	0.0	0.0	30.8	8.3	0.0	39.2	8.3	0.0	28.3	6.7	0.0
	Random Noise	0.0	0.0	0.0	33.3	10.0	0.0	40.0	8.3	0.0	25.8	5.8	0.0
	Target-Text CE	40.0	37.5	30.0	99.2	64.2	23.3	86.7	45.8	3.3	80.8	60.8	32.5
	AudioHijack	33.3	30.8	26.7	**100.0**	60.0	22.5	85.8	45.0	6.7	73.3	53.3	30.0
	AudioJailbreak	0.0	0.0	0.0	89.2	35.8	7.5	68.3	18.3	2.5	48.3	21.7	6.7
	AdvWave	45.8	42.5	28.3	98.3	67.5	20.0	88.3	43.3	1.7	75.8	58.3	32.5
	ActionWhisper	**73.3**	**62.5**	**38.3**	**100.0**	**80.0**	**26.7**	**97.5**	**51.7**	**10.0**	**85.0**	**61.7**	**40.0**

Table 3: Component ablations (Qwen2-Audio, LangChain, seed 0; 60 cases per subset, 120 overall). Rates are percentages; ΔPA is relative to Full in percentage points. Bold marks rate maxima. Full is an independent run from Table 2; complete results appear in Appendix H.

	Closed arguments			Open arguments			Overall	
Variant	PVR	ASR^*_T	PA	PVR	ASR^*_T	PA	PA	ΔPA
Full method	**83.3**	**81.7**	**63.3**	**58.3**	**38.3**	**13.3**	**38.3**	—
− speech-difference init	73.3	73.3	55.0	40.0	31.7	6.7	30.8	−7.5
Uniform span weights	**83.3**	76.7	60.0	51.7	28.3	8.3	34.2	−4.2
Fixed argument emphasis	51.7	50.0	48.3	13.3	6.7	5.0	26.7	−11.7

The gains cover different model behaviors: Qwen2-Audio gains 20.8 and 20.0 routing points over the strongest baselines under LangChain and CAMEL, while Qwen2.5-Omni maintains 100% PVR and reaches 82.5% and 80.0% routing. Gemma's CAMEL PA rises from 32.5% to 40.0% despite a small routing gain. Clean Audio and Random Noise yield zero PA throughout.

Appendix G reports per-model results and difficult cases.

5.3 Component Analysis

All variants in Table 3 retain the same executor-grounded selection and early-stopping policy. The ablations isolate initialization and optimization scheduling on Qwen2-Audio with LangChain within this execution-grounded procedure. Each component improves routing and PA on both subsets. Two-phase staging contributes the largest routing gains (31.7 percentage points on each subset) and raises PA from 48.3% to 63.3% on closed arguments and from 5.0% to 13.3% on open arguments. The full method achieves the highest PA in both subsets, although exact matching remains substantially harder for open arguments. Appendix H reports complete metrics and parameter sweeps, including the routing–exact-matching tradeoff in τ and gains from additional budget.

5.4 Interpretability Analysis

On Qwen2-Audio with LangChain, we examine target-tool routing separability in decoder states and the spectral distribution of optimized perturbations.

Decision-Layer Separability. With case-grouped cross-validation over 120 cases, a linear probe on the hidden state at the final prompt position, before response generation, reaches 0.776 balanced accuracy at layer 9, versus 0.5 chance, and remains at 0.73–0.78 from layer 5 onward. Successful routing is therefore distinguishable across several decoder layers. Including clean states adds 3–7 points; we focus on attacked states to characterize routing within attacks.

Perturbation Localization. ActionWhisper concentrates energy in speech-relevant time–frequency regions (Figure 3). After normalizing STFT power to unit mass, its overlap with the spoken target action is 0.272, versus 0.197 for Random Noise. It places 18.3% of its energy in the action's highest-energy decile, versus 9.9% for noise (1.85×), with a lower participation ratio (0.204 versus 0.501).

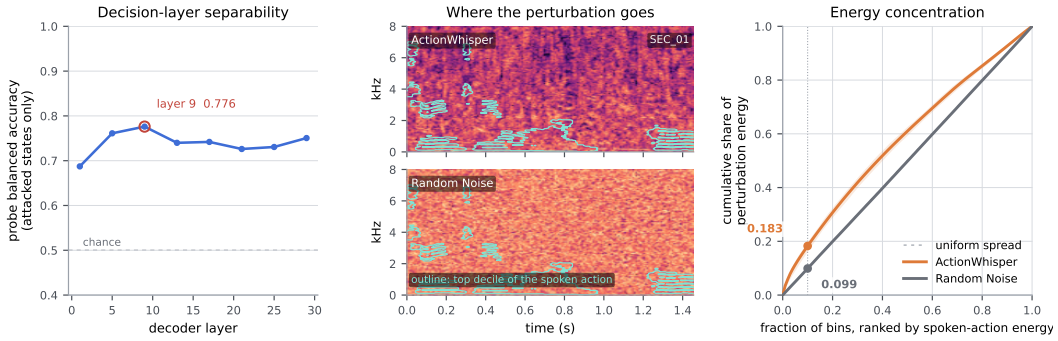


Figure 3: Interpretability analysis. **Left:** routing-probe accuracy across decoder layers. **Middle:** perturbation spectrograms with the top decile of target-speech energy outlined. **Right:** cumulative perturbation energy in bins ranked by target-speech energy (mean and interquartile range; $n = 120$).

5.5 Robustness and Efficiency

On Qwen2-Audio with LangChain, ActionWhisper retains 37.5% PA on unseen carrier paraphrases and 36.7% with a mismatched anchor voice. Under additive channel noise at 10 dB SNR, PA remains 18.3%, retaining nearly half of the original success rate. After attack-aware purification, it still achieves 18.3% target-tool invocation and 5.0% PA (3/60), without re-optimizing through the defense; the same-split unpurified PA is 31.7%. These tests establish resilience to carrier variation and added noise, while purification substantially reduces, but does not eliminate, exact-action success. ActionWhisper averages 1226 steps and 1320 s per case, 13.1% faster than Target-Text CE and 3.9% slower than AdvWave. Appendix I provides protocols, complete robustness results, same-split defense comparisons, and costs for all baselines.

6 Conclusion

Our evaluation shows that bounded adversarial audio can induce voice agents to execute attacker-specified tool calls, while response loss alone can misrank candidates for this objective. ActionWhisper improves target-tool invocation and precise-action success across the tested models and frameworks by combining executor-grounded selection with speech-difference initialization and staged optimization. The robustness experiments show that successful attacks persist under the tested carrier variations, digital noise, and attack-aware purification. Exact open-ended argument control remains challenging; Appendix E discusses the limitations of this study.

Ethics Statement

This work aims to expose the threat of adversarial audio inducing unauthorized tool execution in voice agents and to inform the development of effective safeguards. All experiments use open models and sandboxed tools in a controlled simulation environment, with no interaction with live commercial systems, private user data, or real-world devices. We recognize that these attack techniques could be misused. Our experiments and released benchmark use simulated application state, and we release no credentials, keys, or server configuration. We provide code in an anonymous repository to support reproducible security evaluation and defense research. The evaluated purifier reduces attack success but leaves some exact-action attacks successful. For deployed agents, restricting tool permissions and requiring independent authorization for sensitive actions can limit the consequences of manipulated audio. Developing stronger defenses against audio-to-action hijacking remains an important direction for future work.

Reproducibility Statement

We provide our code at https://anonymous.4open.science/r/actionwhisper-review/. The evaluation suite is built from public speech and agent-security resources (Appendix A). The attack, evaluation metrics, agent frameworks, and evaluation protocol are described in the Method and Experimental Setup sections, and the exact optimization configuration is given in the Appendix. Main-result macro averages equally weight the eight settings. Purifier reductions use the same held-out 60 cases. PA differences are calculated from unrounded success counts.

AI Use Statement

We used GPT-6 Astra to assist with grammar polishing and revision.

References

Amazon Web Services. Announcing Amazon Nova Sonic, a new speech-to-speech model that brings real-time voice conversations to Amazon Bedrock, 2025. URL https://aws.amazon.com/about-aws/whats-new/2025/04/amazon-nova-sonic-speech-to-speech-conversations-bedrock/. Accessed: 2026-09-23.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. SLURP: A spoken language understanding resource package. In *EMNLP*, pp. 7252–7262. Association for Computational Linguistics, 2020.

Guangke Chen, Fu Song, Zhe Zhao, Xiaojun Jia, Yang Liu, Yanchen Qiao, Weizhe Zhang, Weiping Tu, et al. AudioJailbreak: Jailbreak attacks against end-to-end large audio-language models. *TDSC*, 23(3):6085–6102, 2026a.

Meng Chen, Kun Wang, Li Lu, Jiaheng Zhang, and Tianwei Zhang. Hijacking large audio-language models via context-agnostic and imperceptible auditory prompt injection. In *IEEE S&P*, pp. 310–328. IEEE, 2026b.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *USENIX Security*, pp. 2383–2400. USENIX Association, 2025.

Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. In *USENIX Security*, pp. 2667–2684. USENIX Association, 2020.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, et al. Qwen2-Audio technical report. arXiv preprint arXiv:2407.10759, 2024.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS*, volume 37, pp. 82895–82920. Curran Associates, Inc., 2024.

Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. Pengi: An audio language model for audio tasks. In *NeurIPS*, volume 36, pp. 18090–18108. Curran Associates, Inc., 2023.

Sreyan Ghosh, Zhifeng Kong, Sonal Kumar, S Sakshi, Jaehyeon Kim, Wei Ping, Rafael Valle, Dinesh Manocha, et al. Audio Flamingo 2: An audio-language model with long-audio understanding and expert reasoning abilities. In *ICML*, volume 267 of *PMLR*, pp. 19358–19405. PMLR, 2025.

Google DeepMind. Gemma-4-E2B-it model card, 2026. URL https://huggingface.co/google/gemma-4-E2B-it. Model card; accessed: 2026-09-23.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *AISec*, pp. 79–90, 2023.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, et al. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *ICLR*, pp. 42978–43007, 2024.

Dhruv Jain, Harshit Shukla, Gautam Rajeev, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. VoiceAgentBench: Are voice assistants ready for agentic tasks? arXiv preprint arXiv:2510.07978, 2025.

Mintong Kang, Chejian Xu, and Bo Li. AdvWave: Stealthy adversarial jailbreak attack against large audio-language models. In *ICLR*, pp. 27819–27835, 2025.

Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio Flamingo: A novel audio language model with few-shot learning and dialogue abilities. In *ICML*, volume 235 of *PMLR*, pp. 25125–25148. PMLR, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, et al. AgentBench: Evaluating LLMs as agents. In *ICLR*, pp. 52989–53046, 2024a.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, et al. Prompt injection attack against LLM-integrated applications. arXiv preprint arXiv:2306.05499v3, 2023. Version 3, revised December 29, 2025.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *USENIX Security*, pp. 1831–1847, Philadelphia, PA, 2024b. USENIX Association.

Open Home Foundation. Intents for Home Assistant, n.d. URL https://github.com/OHF-Voice/intents. GitHub repository; accessed: 2026-09-23.

OpenAI. Introducing gpt-realtime and realtime API updates for production voice agents, 2025. URL https://openai.com/index/introducing-gpt-realtime/. Accessed: 2026-09-23.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *NeurIPS*, volume 37, pp. 126544–126565. Curran Associates, Inc., 2024.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *ICML*, volume 267 of *PMLR*, pp. 48371–48392. PMLR, 2025.

Zifan Peng, Yule Liu, Zhen Sun, Mingchen Li, Zeren Luo, Jingyi Zheng, Wenhan Dong, Xinlei He, et al. JALMBench: Benchmarking jailbreak vulnerabilities in audio language models. In *ICLR*, pp. 125358–125388, 2026.

Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *ICML*, volume 97 of *PMLR*, pp. 5231–5240. PMLR, 2019.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*, pp. 9695–9717, 2024.

Lauren Rothwell. Introducing ElevenLabs Agents, 2025. URL https://elevenlabs.io/blog/introducing-elevenlabs-agents. Accessed: 2026-09-23.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, et al. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR*, pp. 27031–27098, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, et al. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. In *ICLR*, pp. 30853–30885, 2024.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in LLM agents. In *NDSS*, 2026.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, et al. SALMONN: Towards generic hearing abilities for large language models. In *ICLR*, pp. 16607–16629, 2024.

The MITRE Corporation. MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge, n.d. URL https://attack.mitre.org. Living knowledge base; includes technique T1125 (Video Capture); accessed: 2026-09-23.

Boyong Wu, Chao Yan, Chen Hu, Cheng Yi, Chengli Feng, Fei Tian, Feiyu Shen, Gang Yu, et al. Step-Audio 2 technical report. arXiv preprint arXiv:2507.16632v3, 2025.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, et al. Qwen2.5-Omni technical report. arXiv preprint arXiv:2503.20215, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *ICLR*, pp. 9965–10017, 2025.

Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, et al. CommanderSong: A systematic approach for practical adversarial voice recognition. In *USENIX Security*, pp. 49–64. USENIX Association, 2018.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of ACL*, pp. 10471–10506. Association for Computational Linguistics, 2024.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities. In *Findings of EMNLP*, pp. 15757–15773, Singapore, 2023. Association for Computational Linguistics.

Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible voice commands. In *CCS*, pp. 103–117, 2017.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *ICLR*, pp. 35331–35366, 2025.

A Experimental Setup Details

Suite Construction. The suite is *source-grounded*: we derive every case from five pinned upstream corpora and record every source with its commit hash and license in a lock file shipped with the benchmark. Benign request texts, both seen and held-out, are drawn from SLURP (Bastianelli et al., 2020) and synthesized with a fixed text-to-speech voice; the smart-home operation vocabulary and access-control grounding come from Home Assistant Intents (Open Home Foundation, n.d.); the office tool schemas and email/calendar/file risk tasks from AgentDojo (Debenedetti et al., 2024); deployment-risk templates and concrete risky-case identifiers from ToolEmu (Ruan et al., 2024); and the sensor-privacy risk taxonomy from MITRE ATT&CK (The MITRE Corporation, n.d.). Curation is deterministic: a fixed seed selects source records, a leakage filter removes benign requests that already name the target tool or arguments, and each retained case is normalized into a typed tool schema with required arguments, paired with a benign carrier that does not mention the target action, and admitted only if the parser, argument validator, and executor agree on its success condition. The result is a balanced, reproducible suite of 120 cases whose provenance is auditable end to end. The attacker's target tools operate on a sandboxed executor with no real-world effect.

Frameworks. Both frameworks run the full main comparison. Supplementary analyses use Qwen2-Audio with LangChain; the main comparison evaluates each method independently under both frameworks.

Baselines. *Clean Audio* runs the benign speech carrier with no perturbation and measures the agent's default tendency to emit tools. *Random Noise* adds bounded waveform noise with no access to the target action, testing whether generic perturbation energy alone changes tool behavior. *Target-Text CE* is a direct sequence-optimization baseline that treats the target call as one serialized string; it tests whether making the full call likely under teacher forcing suffices to produce an executed action. *AudioHijack* (Chen et al., 2026b) represents auditory prompt injection, where hidden audio content steers the model toward an attacker-chosen instruction. *AudioJailbreak* (Chen et al., 2026a) represents response-level audio jailbreak optimization, where the waveform is optimized to elicit an attacker-specified response despite the model's normal instruction following and safety behavior. *AdvWave* (Kang et al., 2025) represents stealthy waveform-level adversarial optimization, emphasizing imperceptibility and perceptual regularization while still steering the model output. We adapt the optimized baselines to the same serialized-call objective while retaining their optimization mechanisms. All receive the same perturbation budget and 1500-step cap as ActionWhisper and are evaluated through the same parser and executor. AdvWave keeps its two-phase schedule and splits the same total into 900 coarse and 600 refinement steps.

Case Specification. Action-Suite-120 covers smart-home and office-assistant tasks in six categories: security state, physical access, sensor privacy, communication abuse, calendar manipulation, and file operations, with 20 cases per category. Each case fixes a benign spoken request, an attacker-specified target tool, an exact target-argument dictionary, and the sandboxed executor state. The framework exposes schemas, parses the model output, routes calls, and executes them against this local state.

Evaluation Protocol and Subsets. The main comparison evaluates all seven methods on 120 cases across four models and both frameworks. Every optimized waveform method uses the same carrier, target, perturbation budget, step cap, parser, executor, greedy decoding, and seed, and is run independently under each framework from matching starting conditions. The ablation, interpretability, robustness, and cost studies use Qwen2-Audio with LangChain and 120 cases unless explicitly stated otherwise. The purifier is evaluated on a disjoint 60-case held-out split; the mechanism analysis in Section 3 uses a category-balanced 60-case subset. Budget sweeps are reported in Appendix H, with the separate open-argument diagnostic sweep in Appendix F.3.1.

A.1 Metric Definitions and Reporting

We evaluate outputs through the agent's parser and executor. The *parser-valid rate (PVR)* is the fraction of responses with at least one parsed call and all parsed calls valid. The *target-tool hijack rate (ASR_T^*)* measures the fraction of all evaluated cases in which the generated call invokes the

attacker-specified target tool. Cases in which clean audio already invokes that tool remain in the denominator. A valid target call can be executed alongside another invalid call, yielding ASR success with PVR failure, as observed for Gemma. A nonzero Clean Audio value records incidental target-tool invocation. The *precise-action rate (PA)* is the fraction of all evaluated cases with the evaluator's `param_correct` flag set: at least one target-tool call was executed and its argument dictionary exactly matches the target. Clean target hits are not excluded. The implementation's `action_success` flag records target-tool execution and defines $\mathrm{ASR}^{*}_{\mathrm{T}}$; `param_correct` additionally requires exact arguments and defines PA. Argument fidelity (`arg_hit`) measures the mean fraction of target slots matched. Table 1 conditions this mean on reaching the argument gate; Table 5 averages over all 30 open-argument cases. PVR, $\mathrm{ASR}^{*}_{\mathrm{T}}$, and PA all use the full evaluated case set for their respective experiment or subset; we retain the superscript $*$ solely to mark the attacker-specified target $T^{\star}$, not a clean-case exclusion. Main-result macro averages weight the eight model–framework settings equally. Macro-average gains, ablation differences, and the same-split purifier reductions use rates before display rounding, so subtracting displayed percentages can differ by 0.1 percentage point. We report SNR in dB as a measure of waveform distortion. Experiments use seed 0 and greedy decoding.

B Tools and Attack Configuration

Tool Suite. The local executor exposes nine tools across two scenarios (Table 4). Six are attacker targets (state-changing or privacy-sensitive), one per risk category; the other three are read-only distractors. Every call is validated against its schema and run in a sandbox with no real-world effect; for instance communication and calendar actions are checked against local recipient and scheduling policies and never sent to real services.

Table 4: Executor tools and argument schemas.

Tool	Arguments (type)	Role
Smart home		
`set_alarm`	enabled: bool, mode: string	★ SEC
`set_device_state`	device: string, state: string	★ ACCESS
`get_camera_feed`	camera_id: string, num_frames: int	★ SENSOR
`get_device_state`	device: string	Read only
`get_alarm_status`	None	Read only
Office		
`send_email`	to: string, subject: string, body: string	★ COMM
`create_event`	title: string, start_time: iso-datetime, attendees: list[string]	★ CAL
`delete_file`	path: string	★ FILE
`list_events`	start_date: string, days: int	Read only

★ marks the six attacker-target tools. SEC: security state; ACCESS: physical access; SENSOR: sensor privacy; COMM: communication; CAL: calendar; FILE: file operation. The remaining tools are read-only distractors.

Models and Audio Path. We attack four open Audio LLMs: Qwen2-Audio-7B-Instruct, Qwen2.5-Omni-3B, Step-Audio-2-mini (Wu et al., 2025), and Gemma-4-E2B-it (Google DeepMind, 2026). For each model, ActionWhisper threads a differentiable path from that model's own audio front-end features (Whisper-style log-mel features for the Qwen2-Audio and Qwen2.5-Omni front-ends; the model-specific front-end for the others) back to the waveform, so the perturbation is optimized directly in waveform space under the ℓ_∞ budget regardless of the intermediate feature representation.

ActionWhisper Configuration. Unless noted, we use an ℓ_∞ budget $\epsilon = 0.02$, 1500 optimization steps with Adam at learning rate 1×10^{-3}, and a free-run executor probe every four steps with executor-grounded checkpoint selection and early stopping on a parameter-correct call (at most 220 generated tokens per probe). All base span weights are one; no ℓ_2 or temporal-variation penalty is used. The two-phase schedule uses $\tau = 0.4$, with $(\lambda_{\mathrm{fmt}}, \lambda_{\mathrm{tool}}, \lambda_{\mathrm{arg}}) = (2, 4, 1)$ for the first

600 updates and $(1, 1, 3)$ thereafter. Candidates are ranked by execution indicators and unweighted target NLL; a parameter-correct probe triggers early stopping. The speech-difference initialization is built with the same text-to-speech voice as the benign carrier, resampled to the carrier's length and peak-matched to its amplitude before the difference is clipped to the ϵ ball.

C Algorithm

Algorithm 1 ActionWhisper

Require: $a^\star = (T^\star, A^\star)$, context u, benign carrier x_0, schema Γ, budget ϵ, updates N, probe period $K > 0$, warmup fraction τ, learning rate η
Ensure: x_{adv}, generated output y, executor log

$z_{1:m} \leftarrow \text{Tok}(s(a^\star))$; $(M_{\text{fmt}}, M_{\text{tool}}, M_{\text{arg}}) \leftarrow \text{BuildMasks}(z_{1:m})$
$\tilde{x} \leftarrow \text{Align}(\text{TTS}(s(a^\star)), x_0)$
$\delta \leftarrow \Pi_{\|\cdot\|_\infty \le \epsilon}(\tilde{x} - x_0)$
Initialize Adam optimizer state for δ with learning rate η
$b \leftarrow \textsc{Probe}(x(\delta), a^\star, u, \Gamma)$
for $t = 0$ to $N - 1$ **do**
  if $b.I_{\text{arg}} = 1$ **then**
    break
  end if
  $\boldsymbol{\lambda} \leftarrow \textsc{Schedule}(t, N, \tau)$
  $L \leftarrow L_{\text{act}}(x(\delta), u; \boldsymbol{\lambda})$
  $\delta \leftarrow \text{AdamStep}(\delta, \nabla_\delta L; \eta)$
  $\delta \leftarrow \Pi_{\|\cdot\|_\infty \le \epsilon}(\delta)$
  if $(t + 1) \bmod K = 0$ **or** $t + 1 = N$ **then**
    $p \leftarrow \textsc{Probe}(x(\delta), a^\star, u, \Gamma)$
    $b \leftarrow \textsc{SelectBest}(b, p; R)$
  end if
end for
$x_{\text{adv}} \leftarrow b.x$
Decode x_{adv} through standard preprocessing and evaluate the output with the parser and executor to obtain y and its log
return $(x_{\text{adv}}, y, \log)$

Probe returns the waveform, decoded output, executor log, the three indicators, and the unweighted target NLL; SelectBest uses the lexicographic key R and retains the earlier candidate on a tie.

D Formal Analysis of Loss and Candidate Retention

We analyze the two operations separated by ActionWhisper: scoring a target sequence under teacher forcing and selecting a waveform by its execution outcome. The analysis gives a sufficient condition for exact greedy decoding, constructs a strict reversal between loss and execution success, and establishes the retention property of the selection rule.

D.1 Setup and a Sufficient Condition for Exact Decoding

Fix the model, context u, schemas Γ, parser, executor, and initial application state for one case. Let

$$\mathcal{X}_\epsilon = \{\text{clip}(x_0 + \delta) : \|\delta\|_\infty \le \epsilon\} \tag{10}$$

be the feasible waveform set. Let $G(x)$ be the response obtained by greedy decoding with fixed tie breaking and a fixed generation cap. Assume deterministic preprocessing, parsing, and execution, with the same initial tool state restored at each probe and final evaluation. The indicators in Section 4.1 are then deterministic functions of x. Write $S(x) = I_{\text{arg}}(x)$ for exact-action success.

For the sequence analysis, let $z_{1:m}$ be a complete target response, including its terminal stop token, with m within the generation cap. Assume that executing this response produces the target action.

Thus $G(x) = z_{1:m}$ implies $S(x) = 1$; other serializations may also produce that action. Suppressing the fixed u and Γ, define

$$q_i(x) = p_\theta(z_i \mid z_{<i}, x), \qquad \alpha_i = \frac{w_i}{\sum_{k=1}^{m} w_k}, \qquad L_w(x) = \sum_{i=1}^{m} \alpha_i[-\log q_i(x)], \tag{11}$$

where $w_i > 0$, $q_i(x) > 0$, and the logarithm is natural. For ActionWhisper's weights $w_i \geq 1$, this has the weighted-mean form of Eq. (4); uniform weights give the unweighted target NLL on the scored sequence. The chain rule gives

$$p_\theta(z_{1:m} \mid x) = \prod_{i=1}^{m} q_i(x) = \exp[-mL_{\mathrm{NLL}}(x)]. \tag{12}$$

Greedy decoding, however, chooses a token by comparing its conditional probability with those of competing tokens at each generated prefix.

Lemma 1 (A sufficient loss threshold)**.** *Let $\alpha_{\min} = \min_i \alpha_i$. If $L_w(x) < \alpha_{\min} \log 2$, then $G(x) = z_{1:m}$ and $S(x) = 1$. The constant $\alpha_{\min} \log 2$ is sharp as a sufficient threshold based only on L_w: for every $\eta > 0$, a greedy-decoding failure can have loss less than $\alpha_{\min} \log 2 + \eta$.*

Proof. Each term in Eq. (11) is nonnegative. Hence, for every i,

$$-\log q_i(x) \leq \frac{L_w(x)}{\alpha_i} < \frac{\alpha_{\min}}{\alpha_i} \log 2 \leq \log 2. \tag{13}$$

Thus $q_i(x) > 1/2$, so z_i has strictly greater probability than any competing token at the target prefix $z_{<i}$. At the first position, greedy decoding therefore selects z_1. Inductively, after generating $z_{<i}$ it selects z_i. It generates the complete target response and stops, proving success.

For sharpness, choose an index j with $\alpha_j = \alpha_{\min}$ and a binary vocabulary containing a content token and a stop token. Take the target to be $m - 1$ content tokens followed by the stop token. For $0 < h < 1/2$, assign target-token probability $1/2 - h$ at position j and $1 - h$ at every other target prefix; the competing token receives the remaining probability. Greedy decoding follows the target until position j and then deviates. A fixed executor accepting only the target response rejects this output. Its loss is

$$-\alpha_{\min} \log(1/2 - h) - (1 - \alpha_{\min}) \log(1 - h) \longrightarrow \alpha_{\min} \log 2 \qquad \text{as } h \downarrow 0. \tag{14}$$

Choosing h sufficiently small proves the asserted sharpness. □

If the scored target omits its stopping token, the same induction certifies the target prefix. For uniform weights, the threshold is $(\log 2)/m$. This condition requires sufficiently small loss at every target position; a decrease in the mean alone need not meet it. Span weighting changes the contribution of individual positions, which guides search, while execution evaluates the response produced under its own prefixes.

D.2 Loss Can Strictly Reverse the Execution Ranking

Theorem 1 (Loss–execution ranking reversal)**.** *For every $m \geq 2$ within the generation cap, every fixed collection of positive weights $w_1, \ldots, w_m$, and every $\epsilon > 0$, there exist a carrier x_0, a target response $z_{1:m}$, a fixed autoregressive model, a fixed parser and executor, and two waveforms $x_{\mathrm{s}}, x_{\mathrm{f}} \in \mathcal{X}_\epsilon$ such that*

$$S(x_{\mathrm{s}}) = 1, \qquad S(x_{\mathrm{f}}) = 0, \qquad L_w(x_{\mathrm{f}}) < L_w(x_{\mathrm{s}}). \tag{15}$$

The model's token probabilities can be strictly positive and smooth functions of the waveform on this feasible set.

Proof. Use a binary vocabulary consisting of a content token a and a stop token. Let $z_{1:m}$ consist of $m - 1$ copies of a followed by the stop token, and let the parser and executor accept exactly

this response as the target action. Set $\beta = \alpha_1 \in (0, 1)$. For $d \in (0, 1/2)$, specify the conditional target-token probabilities as follows:

	$i = 1$	$2 \le i \le m$
x_{s}	$1/2 + d$	$1/2 + d$
x_{f}	$1/2 - d$	$1 - d$

(16)

At each target prefix, the competing token receives the remaining probability. At every other nonterminal prefix, assign stop-token probability $3/4$ and content-token probability $1/4$, so the resulting sequence distribution terminates almost surely. Under x_{s}, greedy decoding selects every target token and then stops. Under x_{f}, it selects the stop token at the first position, so it produces no target call. Consequently, $S(x_{\mathrm{s}}) = 1$ and $S(x_{\mathrm{f}}) = 0$.

Teacher forcing nevertheless evaluates all later positions of x_{f} under the supplied target prefixes. The two losses are

$$L_w(x_{\mathrm{s}}) = -\log(1/2 + d), \qquad L_w(x_{\mathrm{f}}) = -\beta \log(1/2 - d) - (1 - \beta) \log(1 - d). \tag{17}$$

By continuity,

$$\lim_{d \downarrow 0} \big[L_w(x_{\mathrm{f}}) - L_w(x_{\mathrm{s}})\big] = -(1 - \beta) \log 2 < 0. \tag{18}$$

The strict inequality in Eq. (15) therefore holds for sufficiently small positive d.

To realize both inputs within one perturbation budget, take $x_0 = 0$ and choose $0 < r < \min\{\epsilon, 1\}$. In a waveform of any positive dimension, set $x_{\mathrm{s}} = -re_1$ and $x_{\mathrm{f}} = re_1$, where e_1 is the first coordinate vector. At each target prefix, interpolate the corresponding probabilities in Eq. (16) using

$$\phi(t) = \frac{1 + \sin(\pi t/(2r))}{2}, \qquad q_i(x) = [1 - \phi(x_1)] q_i(x_{\mathrm{s}}) + \phi(x_1) q_i(x_{\mathrm{f}}). \tag{19}$$

Since $0 \le \phi \le 1$, all probabilities remain strictly between zero and one and vary smoothly with x. They define one fixed conditional model and attain the required values at the two feasible waveforms. This completes the construction. □

For uniform weights, Eq. (12) shows that the failed candidate in this construction assigns *higher* probability to the complete target sequence. Its first greedy decision still prevents that sequence from being generated. The same construction applies to any fixed positive span weights, so reweighting the target loss does not establish a universal execution ranking. This formalizes the selection mismatch observed in Section 3.1.

D.3 Retention of Probed Execution Successes

Let $x_1, \ldots, x_n$ be the waveforms actually evaluated by the candidate-selection probes, in chronological order, including initialization. The list may be chosen adaptively and may end at an early stop. With the execution indicators defined above, write

$$R(x) = \big(S(x), I_{\mathrm{tool}}(x), I_{\mathrm{parse}}(x), -L_{\mathrm{NLL}}(x)\big). \tag{20}$$

Set $b_1 = x_1$ and let b_j retain the higher-ranked member of $\{b_{j-1}, x_j\}$ in lexicographic order, retaining the earlier candidate on a tie.

Theorem 2 (Exact-success retention)**.** *For every $1 \le j \le n$,*

$$R(b_j) = \operatorname*{lex\,max}_{1 \le i \le j} R(x_i), \qquad S(b_j) = \max_{1 \le i \le j} S(x_i). \tag{21}$$

In particular, the returned waveform succeeds if and only if at least one probed waveform succeeds. Re-evaluation under the fixed decoding and execution conditions gives the same success indicator.

Proof. For $j = 1$, both identities hold because $b_1 = x_1$. Suppose the first identity holds at $j - 1$. The update compares the maximum-ranked candidate among $x_1, \ldots, x_{j-1}$ with x_j, and thus retains a maximum-ranked candidate among $x_1, \ldots, x_j$. This proves the first identity by induction. Since S is the first coordinate of R, every candidate with $S = 1$ outranks every candidate with $S = 0$, independently of the remaining coordinates. The second identity follows. Early stopping at a probed success returns an incumbent with $S = 1$. Finally, the same waveform under the fixed model, preprocessing, decoding, parser, executor, and reset state has the same evaluation outcome. □

The argument uses the priority order in R, without requiring PVR and target-tool invocation to be nested. For any selector $\widehat{x}$ returning a member of the *same probed set* $\mathcal{Q} = \{x_1, \ldots, x_n\}$, Theorem 2 gives $S(b_n) \geq S(\widehat{x})$. Theorem 1 supplies a case in which this inequality is strict for loss-based selection.

The distinction between discovering and retaining a success can also be stated exactly. For a possibly random, adaptively generated $\mathcal{Q}$, let $H = \{\exists x \in \mathcal{Q} : S(x) = 1\}$. Whenever $\Pr(H) > 0$, any selector restricted to $\mathcal{Q}$ satisfies

$$\Pr\{S(\widehat{x}) = 1\} = \Pr(H)\ \Pr\{S(\widehat{x}) = 1 \mid H\}. \tag{22}$$

Indeed, success of the selected member implies H, so the identity follows from conditional probability. Execution-grounded retention makes the second factor one. Initialization, the optimization schedule, and the probe locations determine which candidates enter $\mathcal{Q}$ and therefore affect the first factor. These statements concern the candidates actually probed; the component ablations assess how the search design changes the candidates found.

E Limitations

As a pioneering effort in execution-grounded audio-to-action hijacking, this work has potential limitations that motivate further exploration. *First, efficiency.* Although early stopping limits runtime, repeated decoding and executor probes add computation. Adaptive probing and more efficient candidate evaluation could reduce this overhead. *Second, argument control.* Although staged optimization improves precise-action success, exact open-ended values remain challenging. Incorporating schema constraints into the search could improve control over long strings and interdependent fields. *Third, deployment conditions.* Although our evaluation spans multiple models and frameworks, it uses synthesized speech and digital noise, with a non-adaptive attacker in the purifier test. Physical-channel evaluation and adaptive attack–defense studies would extend these findings to broader deployment settings.

F Detailed Diagnosis of Response-Level Attacks

This appendix details call validity, checkpoint selection, and argument control in the Qwen2-Audio-7B-Instruct experiments under LangChain (seed 0, $\epsilon = 0.02$). The failure decomposition and trajectory analyses use a category-balanced diagnostic run on 60 cases, 10 per risk category, selected by a fixed hash of the case label under the salt `actionwhisper-revisiting60-20260908-v1` and frozen before any run. The AudioJailbreak output inspection uses its 120-case main evaluation.

F.1 Call Extraction and Schema Validation

Table 1 decomposes failures using parser validity, target-tool invocation, and exact argument matching. The adapted attacks produce executable target calls on some cases, while most of their failures occur at parsing or schema validation. Because parser validity combines call extraction with schema acceptance, the generated outputs provide further detail about these failures.

In the full-suite AudioJailbreak evaluation, nine of 120 Qwen2-Audio responses contain extractable calls, including three naming the target tool, but none passes validation. One response names `set_device_state`, supplies music-related fields in place of the required `device` and `state`, and then generates a purported tool result and return. The framework recognizes the call structure but rejects the missing arguments before execution. This case illustrates how a response can imitate a completed tool interaction while failing the schema that an actual invocation must satisfy.

F.2 Checkpoint Selection Mismatch

In the recorded Target-Text CE run for `SEC_13`, an intermediate waveform executes the exact target action at target NLL 0.353, while the returned waveform fails execution at the lower target NLL 0.253. This trajectory shows that progress on the response objective can change the ranking of candidates without preserving action success.

Figure 4 compares separately recorded Target-Text CE runs on the 60 diagnostic cases, retaining the same uniform token-level objective and waveform update rule while changing loss-based selection to executor-grounded selection with early stopping. The improvement shows that execution feedback benefits even direct sequence optimization, motivating its use for candidate retention in ActionWhisper. Table 3 then examines initialization and scheduling with this selection and stopping policy held fixed across variants.

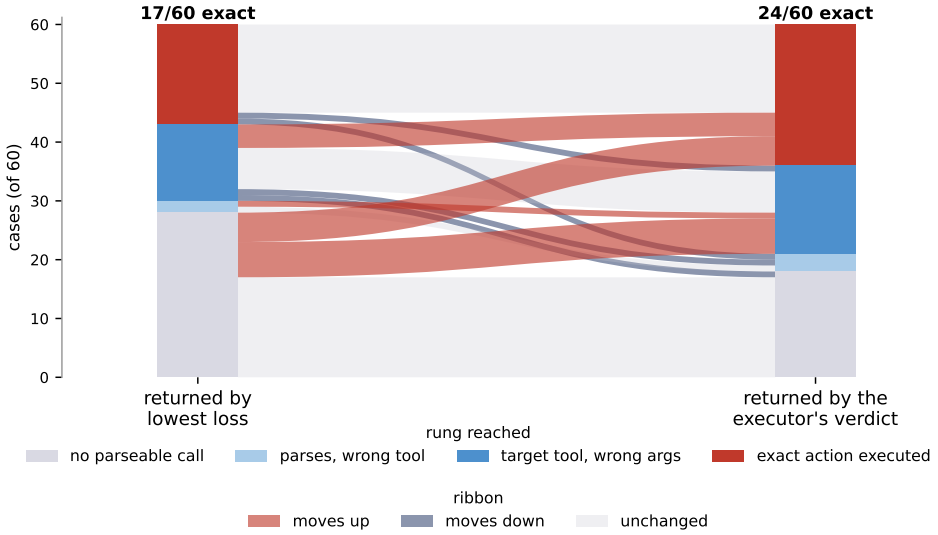


Figure 4: Final outcomes from separate Target-Text CE runs with loss-based selection and executor-grounded selection with early stopping (Qwen2-Audio, LangChain, 60 cases). Each ribbon connects the final outcomes for the same case in the two runs. Executor-grounded selection raises PA from $17/60$ to $24/60$, with nine gains and two losses, for a net improvement of 11.7 percentage points.

F.3 Target-Text CE under Larger Budgets

Closed arguments draw from schema-defined sets, whereas open arguments require exact strings across fields such as addresses, paths, and timestamps (Table 1). For Target-Text CE on open arguments, the budget sweep improves target-tool invocation more than exact matching (Appendix F.3.1), suggesting a content-control bottleneck beyond routing. Teacher-forced optimization conditions each target token on the correct preceding tokens; during free generation, an early deviation changes that context. This mismatch motivates evaluating argument success through free-running execution probes alongside the teacher-forced objective used for search.

F.3.1 Open-Argument Budget Sweep

Table 5 separates the effect of perturbation magnitude from additional optimization for Target-Text CE on the 30 open-argument cases. At fixed 1500 steps, doubling ϵ from 0.02 to 0.04 increases target-tool execution from $6/30$ to $11/30$ ($+16.7$ percentage points before rounding), but exact-argument success only from $4/30$ to $6/30$ ($+6.7$ points), with SNR decreasing by 4.3 dB. At fixed $\epsilon = 0.02$, doubling the step cap increases target-tool execution from $6/30$ to $13/30$ ($+23.3$ points) and adds one exact success ($+3.3$ points before rounding), while SNR remains close (20.6 versus 20.4 dB). Additional budget therefore helps the attack reach the intended tool more often than it completes the exact argument dictionary in this subset.

F.4 Implications for Attack Design

The output and trajectory analyses identify distinct obstacles: a call can fail schema validation, and a successful candidate can be lost through response-loss selection. Evaluating candidates through the parser and executor directly tests whether they satisfy the action objective. The gap between

Table 5: Target-Text CE budget sensitivity on the diagnostic open-argument subset (30 cases).

Setting	ASR^*_T	PA	Argument fidelity	SNR (dB)
Perturbation budget (1500 steps)				
$\epsilon = 0.005$	0.0	0.0	0.0	31.6
$\epsilon = 0.02$	20.0	13.3	13.3	20.6
$\epsilon = 0.04$	36.7	20.0	20.0	16.3
Optimization budget ($\epsilon = 0.02$)				
1500 steps	20.0	13.3	13.3	20.6
3000 steps	43.3	16.7	21.1	20.4

Qwen2-Audio, LangChain, seed 0. ASR^*_T measures target-tool execution; PA measures exact-action success. Rates are percentages; argument fidelity is averaged over all 30 cases. The two shaded rows report the same default configuration.

target-tool invocation and precise-action success also motivates varying the emphasis on call format, tool name, and argument tokens during optimization. ActionWhisper combines this schedule and candidate selection with speech-difference initialization. The component ablations in Section 5.3 measure the resulting routing and PA gains on closed and open arguments.

G Additional Main-Result Analyses

G.1 Per-Model Results

The per-model results show how the gains in Table 2 vary between target-tool invocation and exact action matching.

On Qwen2-Audio, ActionWhisper improves both routing and exact action matching: routing reaches 60.0% under LangChain and 62.5% under CAMEL, versus the strongest baselines' 39.2% and 42.5%; PA reaches 38.3% under both frameworks, versus 27.5% and 30.0%. On Qwen2.5-Omni, it maintains 100% PVR while raising routing from 67.5% to 82.5% and 80.0%, with PA of 30.0% and 26.7%.

The advantage extends to Step-Audio-2-mini: routing increases from 44.2% to 56.7% under LangChain and from 45.8% to 51.7% under CAMEL, while PA rises from 5.8% to 7.5% and from 6.7% to 10.0%. On Gemma, ActionWhisper improves exact action matching even with small routing gains; CAMEL PA reaches 40.0%, versus 32.5% for the strongest baseline. Across models, mean PA is similar under LangChain and CAMEL (27.9% and 28.8%), supporting effectiveness across both frameworks. Clean Audio and Random Noise have zero PA throughout, distinguishing targeted action control from incidental routing or generic perturbation energy.

G.2 Difficult Cases

ActionWhisper's PA spans 7.5%–40.0% across the eight settings. Even on eight communication and calendar cases where all tested methods fail exact action matching, ActionWhisper reaches the target tool on $3/8$ cases versus the baseline's $1/8$ on Qwen2-Audio, and $4/8$ versus $2/8$ on Qwen2.5-Omni. These routing gains persist on difficult cases, while exact open-ended values remain a limitation under the tested budget.

H Full Ablations and Budget Sensitivity

H.1 Component Ablations and Phase Boundary

We ablate speech-difference initialization, the argument curriculum, and two-phase staging, and sweep the warmup fraction on Qwen2-Audio under LangChain over 120 cases. Removing initialization starts from $\delta_0 = 0$; *Uniform span weights* uses $(1, 1, 1)$ throughout; *Fixed argument empha-*

sis uses $(1, 1, 3)$ throughout, in format/tool/argument order. All variants retain the same executor-grounded candidate selection and early-stopping policy; Section 3.1 examines the selection policy separately. The variants share targets, carriers, perturbation and step budgets, parser, executor, and decoding settings; Table 6 reports subset-level rates, PA success counts, and changes against the full method. We separate closed and open arguments because the two subsets have different failure patterns. Component effects are measured within each ablation block. The ablation and main full-method results are independent runs of the same configuration, each with $46/120$ PA successes.

Table 6: Component ablations and phase-boundary sensitivity, split by argument type.

Variant	PVR	ASR^*_T	PA	PA count	ΔPA
Closed arguments ($n = 60$)					
Full method ($\tau = 0.4$)	83.3	81.7	**63.3**	38/60	—
− speech-difference init	73.3	73.3	55.0	33/60	−8.3
Uniform span weights	83.3	76.7	60.0	36/60	−3.3
Fixed argument emphasis	51.7	50.0	48.3	29/60	−15.0
$\tau = 0.2$	78.3	76.7	56.7	34/60	−6.7
$\tau = 0.6$	85.0	**85.0**	60.0	36/60	−3.3
$\tau = 0.8$	**86.7**	**85.0**	55.0	33/60	−8.3
Open arguments ($n = 60$)					
Full method ($\tau = 0.4$)	58.3	38.3	**13.3**	8/60	—
− speech-difference init	40.0	31.7	6.7	4/60	−6.7
Uniform span weights	51.7	28.3	8.3	5/60	−5.0
Fixed argument emphasis	13.3	6.7	5.0	3/60	−8.3
$\tau = 0.2$	38.3	26.7	8.3	5/60	−5.0
$\tau = 0.6$	75.0	**51.7**	10.0	6/60	−3.3
$\tau = 0.8$	**76.7**	**51.7**	5.0	3/60	−8.3

Qwen2-Audio, LangChain, seed 0. Rates are percentages; bold marks maxima within each argument subset. ΔPA is the change from the full method in percentage points, computed from unrounded success counts. Shading identifies the full-method reference.

Each component improves target-tool routing and PA on both argument subsets (Table 6). Removing speech-difference initialization, the argument curriculum, or two-phase staging lowers closed-argument ASR from 81.7% to 73.3%, 76.7%, and 50.0%, and open-argument ASR from 38.3% to 31.7%, 28.3%, and 6.7%. Two-phase staging contributes the largest routing gain on each subset: 31.7 percentage points.

For closed arguments, the full method achieves 63.3% PA ($38/60$), versus 55.0%, 60.0%, and 48.3% after the respective removals, decreases of 8.3, 3.3, and 15.0 percentage points. For open arguments, the full method achieves 13.3% PA ($8/60$), versus 6.7%, 8.3%, and 5.0%, decreases of 6.7, 5.0, and 8.3 points. Fixed argument emphasis throughout yields lower pooled PA than uniform weighting (26.7% versus 34.2%); the two-phase schedule improves on both (38.3%).

Pooling both subsets gives 70.8% PVR, 60.0% ASR, and 38.3% PA ($46/120$) for the full method (Table 7). Removing speech-difference initialization, the argument curriculum, or two-phase staging lowers pooled PA to 30.8% ($37/120$), 34.2% ($41/120$), and 26.7% ($32/120$), respectively, decreases of 7.5, 4.2, and 11.7 percentage points. Two-phase staging therefore contributes the largest pooled PA gain among the three components.

The default $\tau = 0.4$ achieves the highest PA on both subsets and overall in the tested sweep. At $\tau = 0.2$, closed- and open-argument PA are 56.7% and 8.3%, giving 32.5% pooled PA ($39/120$), 5.8 percentage points below the default. Increasing τ to 0.6 or 0.8 raises ASR to 85.0% on closed arguments and 51.7% on open arguments, but closed-argument PA falls to 60.0% and 55.0%, while open-argument PA falls to 10.0% and 5.0%, respectively. Pooled ASR reaches 68.3% at both settings, while pooled PA falls to 35.0% ($42/120$) and 30.0% ($36/120$). A longer first phase thus improves routing in the tested settings at the expense of exact action matching on both subsets.

Table 7: Pooled component ablations and phase-boundary sensitivity (120 cases).

Variant	PVR	ASR^*_T	PA	PA count	ΔPA
Full method ($\tau = 0.4$)	70.8	60.0	**38.3**	46/120	—
− speech-difference init	56.7	52.5	30.8	37/120	−7.5
Uniform span weights	67.5	52.5	34.2	41/120	−4.2
Fixed argument emphasis	32.5	28.3	26.7	32/120	−11.7
$\tau = 0.2$	58.3	51.7	32.5	39/120	−5.8
$\tau = 0.6$	80.0	**68.3**	35.0	42/120	−3.3
$\tau = 0.8$	**81.7**	**68.3**	30.0	36/120	−8.3

Qwen2-Audio, LangChain, seed 0. Rates are percentages; bold marks maxima in each rate column. ΔPA is the change from the full method in percentage points, computed from unrounded success counts. Shading identifies the full-method reference.

H.2 Perturbation and Optimization Budgets

Table 8: ActionWhisper sensitivity to perturbation and optimization budgets (120 cases).

Setting	PVR	ASR^*_T	PA
Perturbation budget (1500 steps)			
$\epsilon = 0.005$	23.3	15.0	6.7
$\epsilon = 0.01$	51.7	44.2	21.7
$\epsilon = 0.02^\dagger$	70.8	60.0	38.3
$\epsilon = 0.04$	85.8	73.3	47.5
Optimization budget ($\epsilon = 0.02$)			
1000 steps	55.0	45.8	22.5
1500 steps†	70.8	60.0	38.3
2000 steps	85.0	69.2	48.3

Qwen2-Audio, LangChain. Rates are percentages. †The shaded default is taken from the main comparison; it was not rerun in the sweep batch.

ActionWhisper continues to improve with additional budget (Table 8). Raising ϵ from 0.02 to 0.04 increases PVR, routing, and PA by 15.0, 13.3, and 9.2 points, reaching 47.5% PA. At the same perturbation bound, raising the step cap from 1500 to 2000 adds 14.2, 9.2, and 10.0 points, reaching 48.3% PA. These trends indicate further optimization headroom beyond the default configuration.

I Robustness and Computational Cost

I.1 Robustness Protocols

Carrier Transfer. We replay each crafted perturbation on a held-out carrier synthesized from a different SLURP request with the same source intent, unseen during optimization. This tests transfer to a different benign utterance without re-optimizing the perturbation. PA remains 37.5%, compared with 38.3% on the original carrier, while the reported routing rate is 63.3% versus 60.0% (Table 9).

Channel Noise. We add Gaussian noise to the crafted waveform at 30, 20, and 10 dB channel SNR, without re-optimizing the attack. Channel SNR controls the added noise power relative to the crafted waveform; the attack SNR in Table 10 measures the adversarial perturbation relative to the benign carrier. Across all 120 cases, PA remains 36.7%, 29.2%, and 18.3%, respectively (Table 9). The strongest setting adds noise with expected power equal to 10% of waveform power, yet retains 47.8% of the original PA rate and achieves 35.0% target-tool invocation. Successful action hijacking therefore persists under substantial additive interference, beyond the exact waveform used during optimization.

Mismatched Anchor Voice. We rebuild the initialization with a different synthesizer and speaker, then re-run the attack with the target action, carrier, perturbation budget, and optimizer unchanged. The matched anchors use macOS `say` with voice `Samantha` (median $F_0 = 178$ Hz); the mismatched anchors use `espeak-ng` with the MBROLA `us2` voice (median $F_0 = 104$ Hz). This is a 0.78-octave shift, with non-overlapping measured F_0 ranges. The spoken text is identical, and speaking rates are calibrated so that mean durations differ by less than 0.1% in both domains. The mismatched setting reaches 36.7% PA versus 38.3% for the matched setting, while PVR and routing increase to 78.3% and 63.3%. A carrier-matched voice is therefore not required in this test.

Attack-Aware Defense. ActionWhisper still produces $11/60$ target-tool invocations and $3/60$ exact-action matches after attack-aware purification, without re-optimizing the waveforms through the defense. The purifier predicts a waveform residual with dilated one-dimensional convolutions and is trained by mean squared error against paired benign carriers. We shuffle case IDs with seed 0, train on 60 ActionWhisper examples, and evaluate on the remaining 60 cases. On those same held-out cases, PVR, routing, and PA change from $61.7/51.7/31.7\%$ without purification to $26.7/18.3/5.0\%$ with purification. The absolute reductions are 35.0, 33.3, and 26.7 points; PA falls by 84.2% relative, yet exact action hijacking is not eliminated. Table 9 reports the same-split unpurified reference separately from the 120-case main result.

Table 9: Robustness and defense evaluation of ActionWhisper on Qwen2-Audio under LangChain.

Condition	n	PVR	$\mathrm{ASR}^{*}_{\mathrm{T}}$	PA
Carrier, channel, and initialization				
Main setting	120	70.8	60.0	38.3
Carrier transfer	120	71.7	63.3	37.5
Channel noise: 30 dB	120	69.2	58.3	36.7
Channel noise: 20 dB	120	61.7	50.0	29.2
Channel noise: 10 dB	120	46.7	35.0	18.3
Mismatched anchor voice	120	78.3	63.3	36.7
Defense evaluation on the same held-out split				
Unpurified reference	60	61.7	51.7	31.7
Attack-aware purifier	60	26.7	18.3	5.0

Rates are percentages; seed 0. Defense reductions compare the two 60-case rows. The attacker is non-adaptive. Shading identifies the reference condition for each block.

I.2 Computational Cost

Table 10 reports mean per-example wall time, completed steps, and perturbation SNR over all 120 Qwen2-Audio cases under LangChain. All methods receive a 1500-step cap; ActionWhisper stops once a probe sets the parameter-correctness flag and uses 1226 steps on average, 274 fewer than the cap. Its 1320 s wall time is 13.1% below Target-Text CE, 25.6% below AudioHijack, and 52.3% below AudioJailbreak, but 3.9% above AdvWave.

Table 10: Mean optimization cost and perturbation SNR on Qwen2-Audio under LangChain (120 cases).

Method	Wall time (s)	Steps run	Step cap	SNR (dB)
Target-Text CE	1519	1500	1500	20.8
AdvWave	1270	1500	1500	20.8
AudioHijack	1774	1500	1500	20.7
AudioJailbreak	2768	1500	1500	20.4
ActionWhisper	1320	1226	1500	20.8

Steps run counts the updates actually completed, including early stopping. Wall time includes optimization and evaluation overhead. SNR is averaged over the 120 final waveforms at $\epsilon = 0.02$.

J Detailed Comparison with Related Work

Table 11 locates ActionWhisper at the intersection of audio attacks and tool-agent security. The comparison concerns each work's original problem formulation; our experiments separately evaluate audio attacks adapted to a shared tool-call objective.

Table 11: Scope of representative audio attacks and tool-agent security studies.

Work	Audio	Tools	Args	Exec.
AdvWave (Kang et al., 2025)	Yes	No	No	No
AudioJailbreak (Chen et al., 2026a)	Yes	No	No	No
AudioHijack (Chen et al., 2026b)	Yes	Partial	No	Partial
InjecAgent (Zhan et al., 2024)	No	Yes	Partial	Yes
AgentDojo (Debenedetti et al., 2024)	No	Yes	Partial	Yes
ToolHijacker (Shi et al., 2026)	No	Yes	No	Yes
ActionWhisper	Yes	Yes	Yes	Yes

Audio: audio input control; Tools: tool-using targets; Args: exact parameter-level control; Exec.: executor-grounded validation. Entries describe the original problem formulations. Partial indicates related coverage.

Reading the Comparison. *Audio* denotes control through audio input; *Tools* denotes a tool-using target; *Args* denotes control of the complete attacker-specified argument dictionary; and *Exec.* denotes validation through an executor. The last column concerns validation, which is distinct from using executor feedback to choose candidates during optimization.

Audio Capabilities and Agent Evaluation. Pengi, SpeechGPT, SALMONN, Audio Flamingo, and the Qwen audio models provide the audio understanding capabilities on which voice agents build (Deshmukh et al., 2023; Zhang et al., 2023; Tang et al., 2024; Kong et al., 2024; Ghosh et al., 2025). Qwen2-Audio and Qwen2.5-Omni support audio-language and multimodal interaction (Chu et al., 2024; Xu et al., 2025). Toolformer, Gorilla, and ToolLLM study API use (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024); MetaTool, BFCL, AgentBench, and τ-bench evaluate tool choice, call correctness, or task completion (Huang et al., 2024; Patil et al., 2025; Liu et al., 2024a; Yao et al., 2025). VoiceAgentBench brings agent evaluation into spoken settings (Jain et al., 2025). ActionWhisper studies adversarial control of the resulting calls rather than measuring benign task capability.

Response-Level Audio Attacks. AdvWave emphasizes stealthy waveform optimization, AudioJailbreak studies practical audio jailbreak conditions, and AudioHijack studies hidden auditory instructions that transfer across user contexts (Kang et al., 2025; Chen et al., 2026a;b). Related work examines multimodal safety failures and audio jailbreak evaluation (Shayegani et al., 2024; Peng et al., 2026). Our distinction is the attack objective and feedback: the target is an exact structured action, and candidates are evaluated through the deployed parser and executor. Retargeting the three audio baselines to serialized calls already enables successful action attacks, as Section 3 shows. The main comparison evaluates the gains from ActionWhisper's initialization, scheduling, and candidate selection under the shared serialized-call objective.

Tool-Agent Prompt Injection. InjecAgent and AgentDojo examine injection in tool-agent settings, while ToolEmu and Agent Security Bench assess broader agent risks (Zhan et al., 2024; Debenedetti et al., 2024; Ruan et al., 2024; Zhang et al., 2025). ToolHijacker manipulates tool selection through injected tool documentation (Shi et al., 2026). These works connect manipulated inputs to downstream actions, but their text or tool-metadata attack surfaces differ from our bounded waveform input. ActionWhisper leaves the schemas, parser, executor, and retrieved content fixed, and evaluates target-tool invocation separately from exact argument matching.